


Unidade VI:

Algoritmo de Ordenação por Seleção



PUC Minas

Adaptação dos slides elaborados pelo Instituto de Ciências Exatas e
Informática - Departamento de Ciência da Computação

- **Introdução sobre Ordenação Interna** 
- Funcionamento básico
- Algoritmo
- Análise do número de movimentações e comparações
- Conclusão

Introdução sobre Ordenação Interna

- Muitas aplicações requerem dados de forma ordenada
- Entrada: *array* com n elementos
- A ordenação é dita **Interna** quando a lista de elementos cabe na memória principal, caso contrário, é dita **Externa**
- Chave de Pesquisa: Atributo utilizado para ordenar os registros
- Curiosidade: [Sorting Algorithms Animations | Toptal®](#)

Análise dos Algoritmos de Ordenação Interna

- Operações fundamentais: **comparação** e **movimentação** entre elementos do *array*
- A complexidade ótima para a ordenação interna em número de comparações do pior e do caso médio é $\Theta(n \times \log(n))$
- Vários algoritmos de ordenação interna alcançam esse limite

Algoritmos Estáveis vs. Não Estáveis

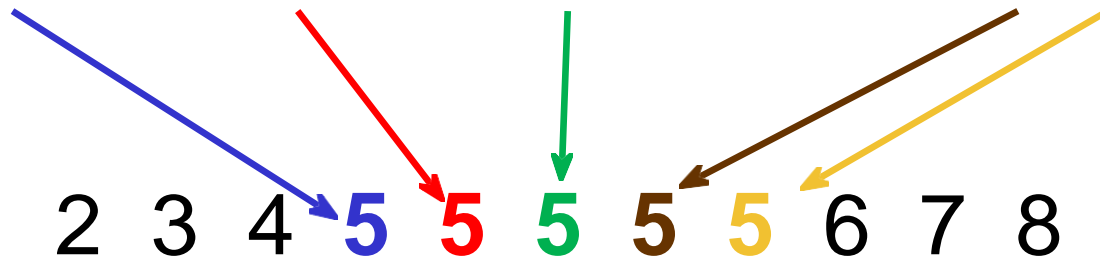
- Um algoritmo é dito estável se depois da execução, os elementos com a mesma chave mantiverem a ordem relativa entre as chaves repetidas
- No exemplo abaixo, a ordem dos elementos azul, vermelho, verde e marrom e amarelo é a mesma

- Antes:

9 5 1 4 5 0 7 5 2 8 6 3 5 5

- Depois:

0 1 2 3 4 5 5 5 5 5 6 7 8 9



Algoritmos Estáveis vs. Não Estáveis

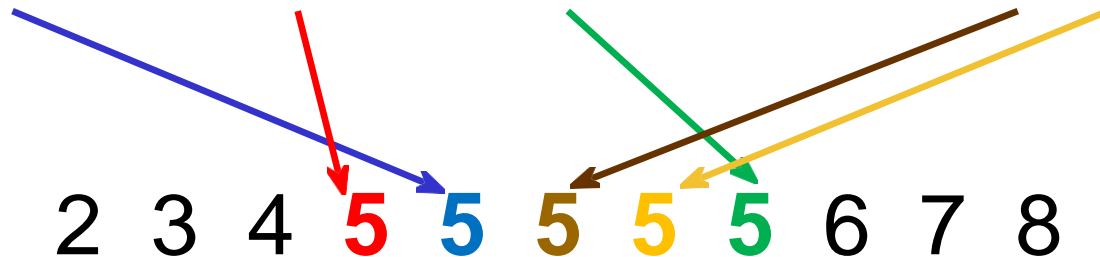
- Um algoritmo é dito estável se depois da execução, os elementos com a mesma chave mantiverem a ordem relativa entre as chaves repetidas
- No exemplo abaixo, a ordem dos elementos azul, vermelho, verde e marrom e amarelo não foi mantida (algoritmo não estável)

- Antes:

9 5 1 4 5 0 7 5 2 8 6 3 5 5

- Depois:

0 1 2 3 4 5 5 5 5 5 6 7 8 9




Algoritmos Adaptáveis X Não Adaptáveis

- Um algoritmo é dito adaptável se ele aproveita uma ordenação prévia dos elementos do array;
 - Quando o algoritmo é adaptável, ele evita movimentações desnecessárias, já que os elementos se encontram em suas posições corretas;
 - Já quando o algoritmo não é adaptável, mesmo os elementos estando na posição correta, ainda assim é realizada a movimentação dos dados;

Complexidade de memória

- Mais um fator relevante é a memória extra que algoritmos de ordenação utilizam para conseguir ordenar o arranjo;
- Os métodos de ordenação que não precisam de memória extra são chamados de métodos in place (ou in situ);
- Basicamente, considera-se memória extra, a necessidade de um array adicional para efetuar a ordenação dos elementos do array;

- Introdução sobre Ordenação Interna
- **Funcionamento básico** 
- Algoritmo
- Análise do número de movimentações e comparações
- Conclusão

Funcionamento Básico

- Procure o menor elemento do *array*
- Troque a posição do menor elemento com o primeiro
- Volte ao primeiro passo e considere o *array* a partir da próxima posição

Exemplo

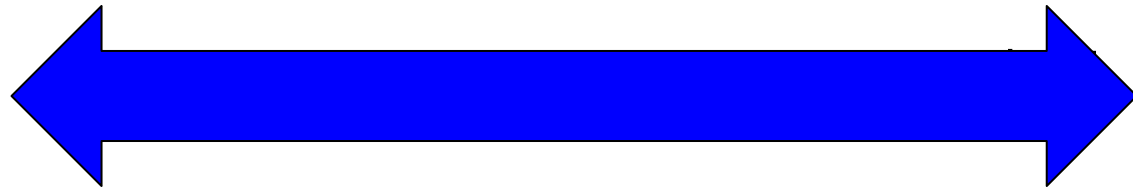
Legenda: - menor elemento em vermelho
- parte ordenada está de azul

101 115 30 63 47 20

101 115 30 63 47 20

Menor
elemento

101



20

Trocando a posição do menor
elemento com o primeiro

Parte
ordenada

20

115 30 63 47 101

Parte a ser ordenada

20 115 30 63 47 101

20 115 30 63 47 101

Menor
elemento

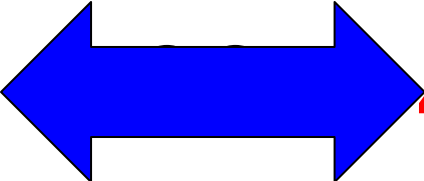
20 115  30 63 47 101

20 30 115 63 47 101

Trocando a posição do menor
elemento com o primeiro

20 30 115 63 47 101

Menor
elemento

20 30 115  47 101

20 30 47 63 115 101

Trocando a posição do menor
elemento com o primeiro

20 30 47 63 115 101

Menor
elemento

20 30 47 63 115 101

Trocando a posição do menor
elemento com o primeiro

20 30 47 63 115 101

Menor
elemento

20 30 47 63 115 101




20 30 47 63 101 115

Trocando a posição do menor
elemento com o primeiro


20 30 47 63 101 115

O algoritmo terminou? Por que?

- Introdução sobre Ordenação Interna
- Funcionamento básico
- **Algoritmo** 
- Análise do número de movimentações e comparações
- Conclusão

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```



(Obs.1): No Seleção, os valores de i serão: 0, 1, 2, 3, ... e $(n-2)$

O laço externo não é executado quando i igual a $(n-1)$

101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo

2

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

(Obs.2): i endereça a posição do elemento a ser inserido no conjunto ordenado

101

115

30

63

47

20

0

1

2

3

4

5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    3 → for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

(Obs.3): O laço interno procura a posição do menor elemento no conjunto a ser ordenado

101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = 4  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

(Obs.4): j começa na primeira posição a ser comparada com a posição menor

101	115	30	63	47	20
0	1	2	3	4	5

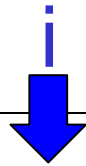
Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```



101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

true: $0 < 5$ 

101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

↓ menor
↓ i

101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

↓ menor
↓ i

↓ j

101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

true: $1 < 6$

↓ menor

↓ i

↓ j

101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

false: 101 > 115

↓ menor
↓ i

↓ j

101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

↓ menor

↓ i

↓ j

101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

true: $2 < 6$

↓ menor
↓ i

↓ j

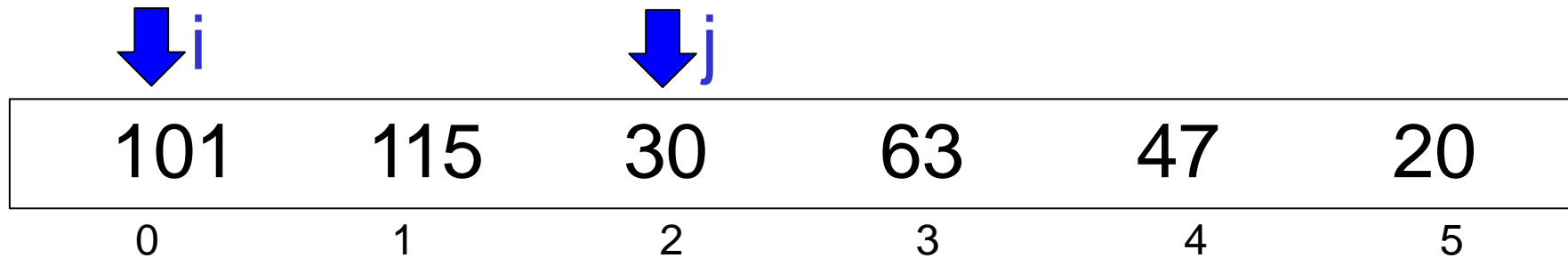
101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo

```

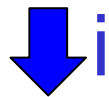
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

true: $101 > 30$



Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```



i



menor



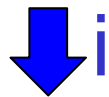
j

101	115	30	63	47	20
0	1	2	3	4	5

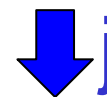
Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

menor



i



j

101	115	30	63	47	20
0	1	2	3	4	5

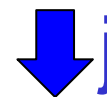
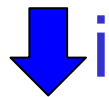
Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

true: $3 < 6$

menor



101	115	30	63	47	20
0	1	2	3	4	5

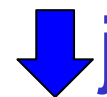
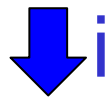
Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

false: $30 > 63$

menor

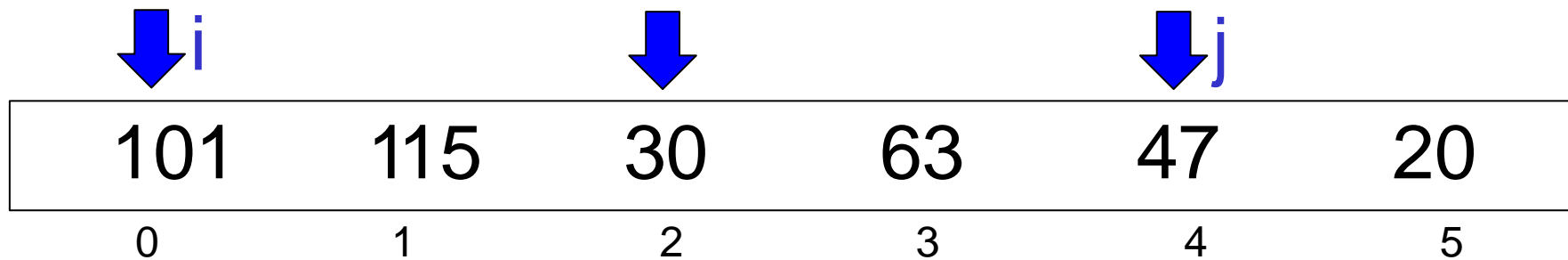


101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

menor



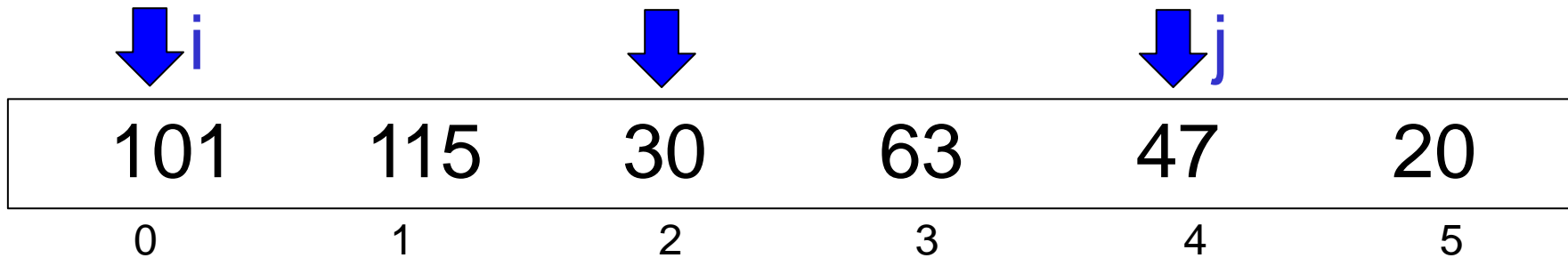
Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

true: $4 < 6$

menor



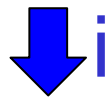
Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

false: $30 > 47$

menor



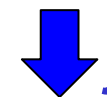
101

0



30

2



47

4

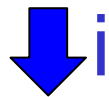
20

5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

menor



101

0



30

2



20

5

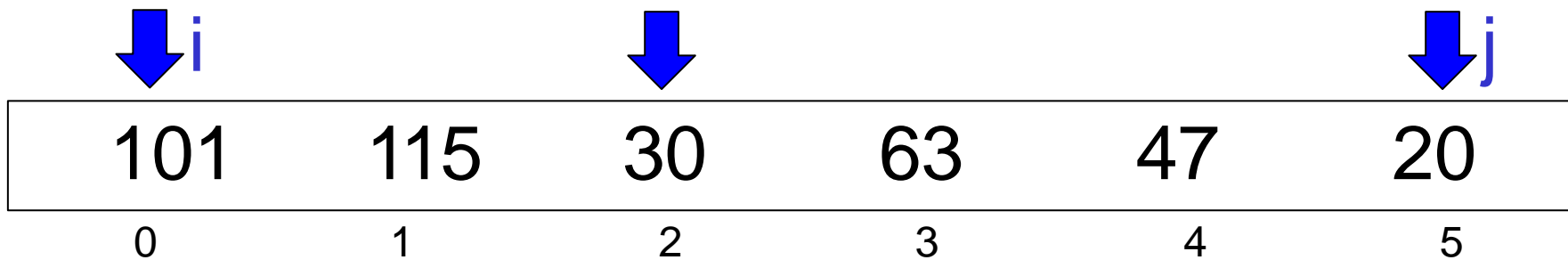
Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

true: $5 < 6$

menor



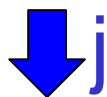
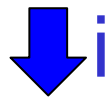
Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

true: $30 > 20$

menor



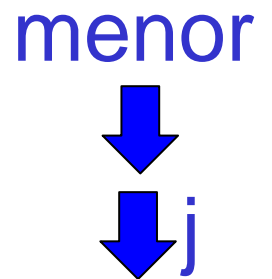
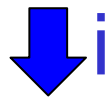
101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

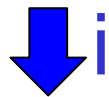
true: $30 > 20$



101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```



101

0

115

1

30

2

63

3

47

4

20

5

menor



j

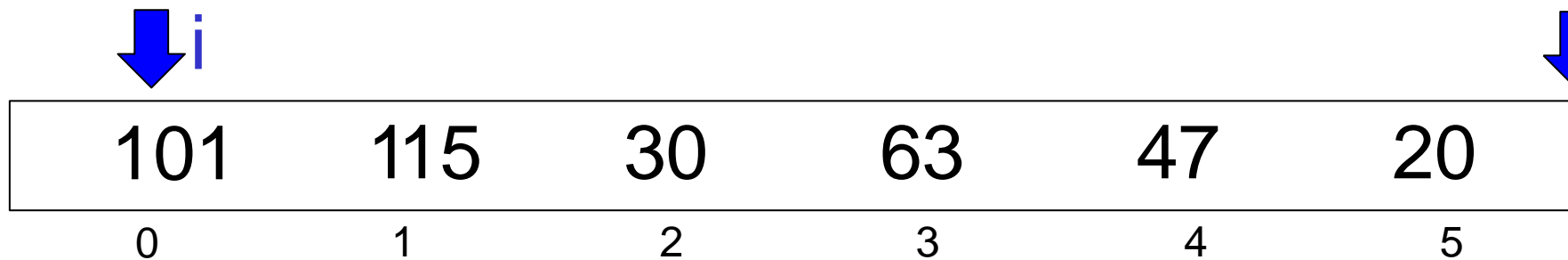


Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

false: $6 < 6$

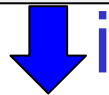


Algoritmo

```
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
}
```

```
int temp = array[menor];
array[menor] = array[i];
array[i] = temp;
```

```
}
```



20

115

30

63

47

101

0

1

2

3

4

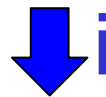
5

menor



Algoritmo

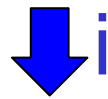
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```



20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

true: $1 < 5$ 

20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

↓ menor
↓ i

20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

↓ menor
↓ i ↓ j

20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

true: $2 < 6$

↓ menor
↓ i ↓ j

20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

true: $115 > 30$

↓ menor
↓ i ↓ j

20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

true: $115 > 30$

↓ menor

↓ i

↓ j

20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

↓ menor

↓ i

↓ j

20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
```

true: $3 < 6$

↓ menor

↓ i

↓ j

20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

false: $30 > 63$

↓ menor

↓ i

↓ j

20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

↓ menor

↓ i

↓ j

20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

true: $4 < 6$

↓ menor

↓ i

↓ j

20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

false: $30 > 47$

↓ menor

↓ i

↓ j

20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

↓ menor

↓ i

↓ j

20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
```

true: $5 < 6$

↓ menor

↓ i

↓ j

20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

false: $30 > 101$

↓ menor

↓ i

↓ j

20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

↓ menor

↓ i

↓ j

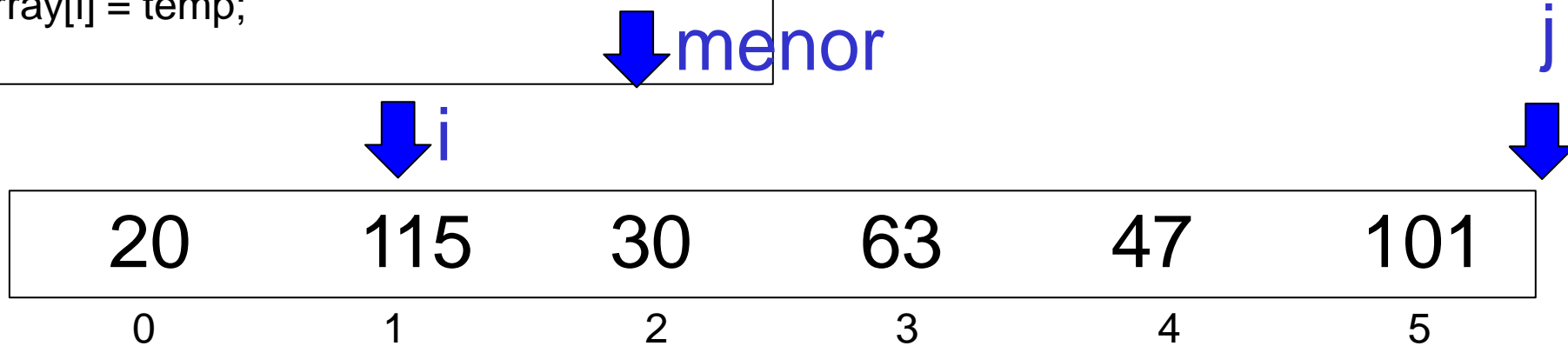
20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

false: $6 < 6$



Algoritmo

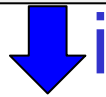
```
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
}
```

```
int temp = array[menor];
array[menor] = array[i];
array[i] = temp;
```

```
}
```



menor



i



j

20	30	115	63	47	101
----	----	-----	----	----	-----

0

1

2

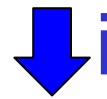
3

4

5

Algoritmo

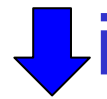
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```



20	30	115	63	47	101
0	1	2	3	4	5

Algoritmo

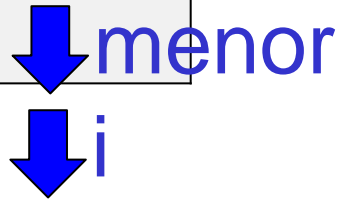
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

true: $2 < 5$ 

20	30	115	63	47	101
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```



20	30	115	63	47	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

↓ menor
↓ i ↓ j

20	30	115	63	47	101
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
```

true: $3 < 6$

↓ menor
↓ i ↓ j

20	30	115	63	47	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

true: $115 > 63$

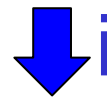
↓ menor
↓ i ↓ j

20	30	115	63	47	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```



i



menor

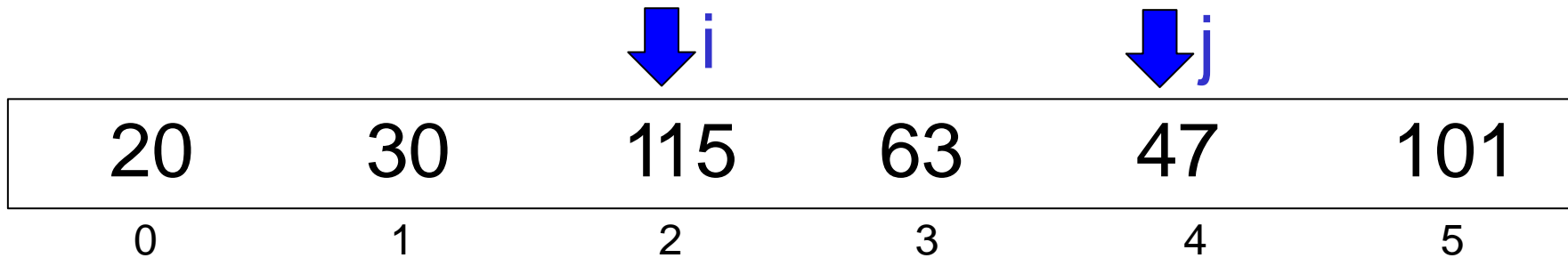


j

20	30	115	63	47	101
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```



Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

true: $4 < 6$

↓ menor

↓ j

↓ i

20	30	115	63	47	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

true: $63 > 47$

↓ menor

↓ i

↓ j

20	30	115	63	47	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

true: $63 > 47$

↓ menor
↓ j

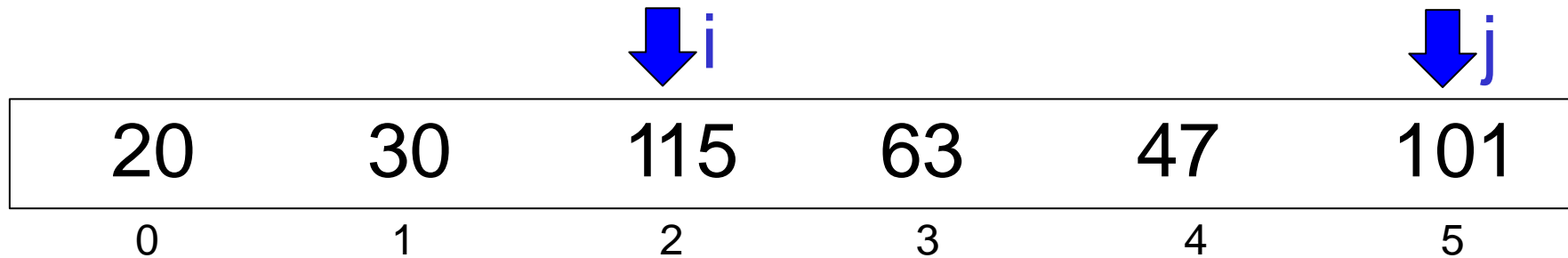
↓ i

20	30	115	63	47	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```



Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

true: $5 < 6$

↓ menor

↓ i

↓ j

20	30	115	63	47	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

false: $47 > 101$

↓ menor

↓ i

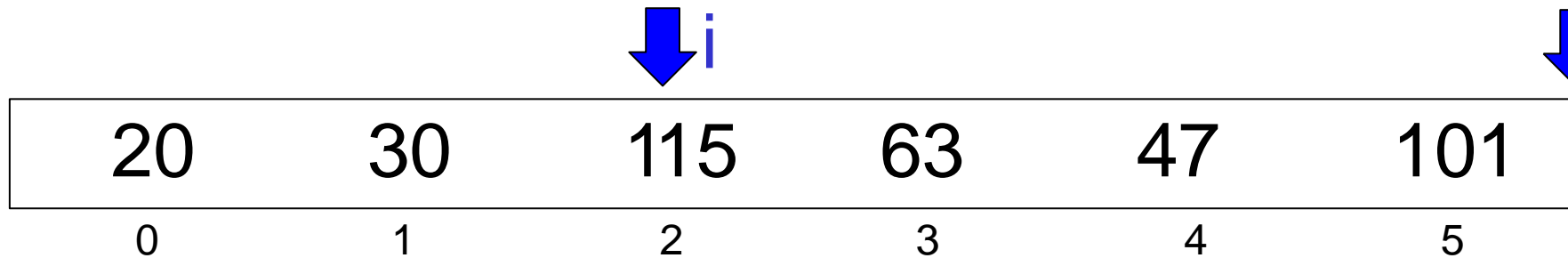
↓ j

20	30	115	63	47	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

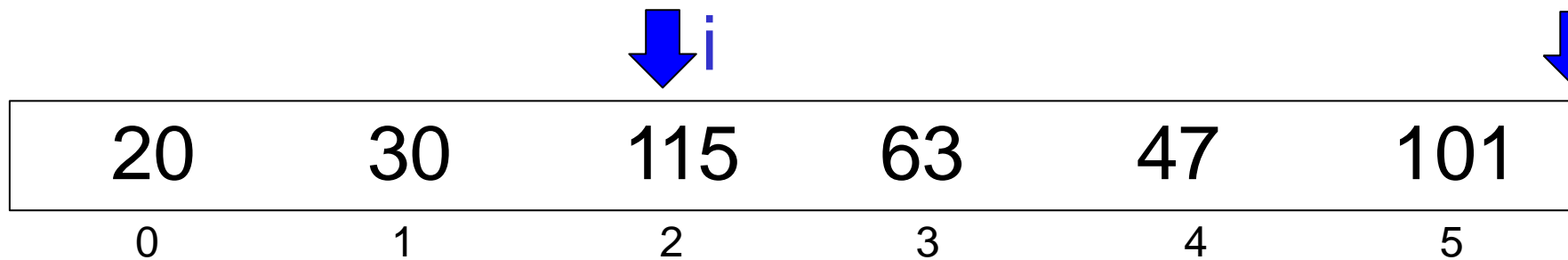


Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

false: $6 < 6$



Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```


↓ menor

↓ i

20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```




20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

true: $3 < 5$



20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

↓ menor
↓ i

20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

↓ menor
↓ i ↓ j

20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

true: $4 < 6$

↓ menor

↓ i

↓ j

20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```


false: $63 > 115$

↓ menor
↓ i ↓ j

20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```



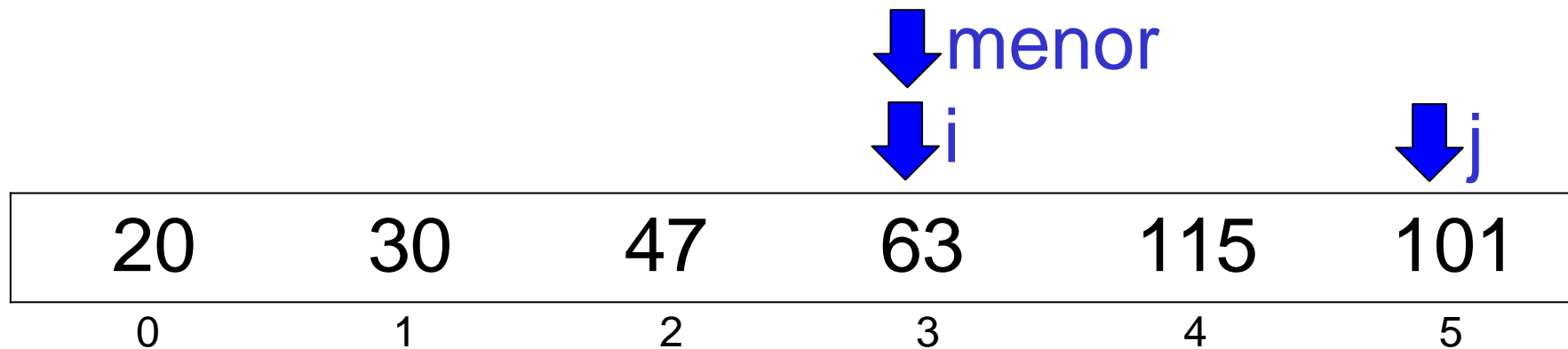
20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

true: $5 < 6$

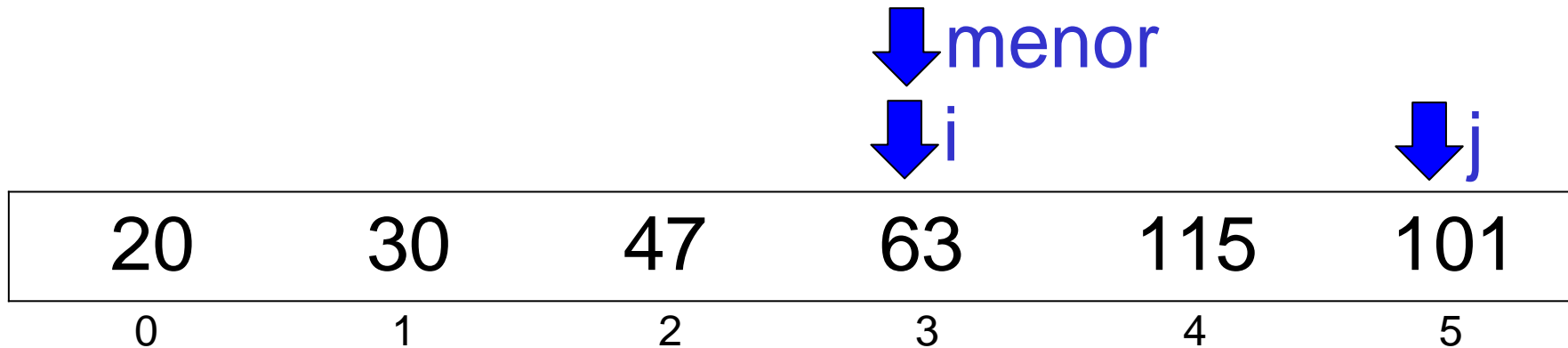


Algoritmo

```


for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

false: $63 > 101$



Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```



20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

false: $6 < 6$

20	30	47	63	115	101
0	1	2	3	4	5

↓ menor
↓ i

j
↓

Algoritmo

```

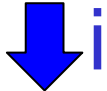
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

↓ menor
↓ i

20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo

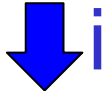
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```



20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

true: $4 < 5$ 

20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

↓ menor
↓ i

20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

↓ menor
↓ i ↓ j

20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

true: $5 < 6$

↓ menor
↓ i ↓ j

20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

true: $115 > 101$

↓ menor
↓ i ↓ j

20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo

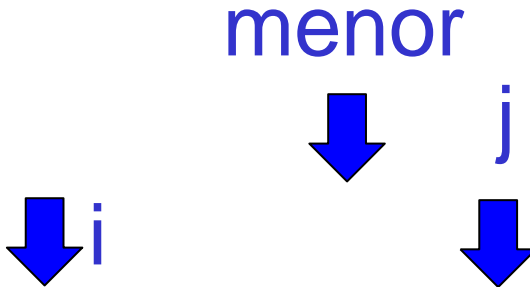
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

20	30	47	63	115	101
0	1	2	3	4	5

Diagram illustrating the selection sort process on an array. The array contains the values 20, 30, 47, 63, 115, and 101 at indices 0 through 5. A blue arrow labeled 'i' points to index 4 (value 115). Another blue arrow labeled 'menor' points to index 5 (value 101), indicating that 101 is the minimum element found in the current iteration's subarray.

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```



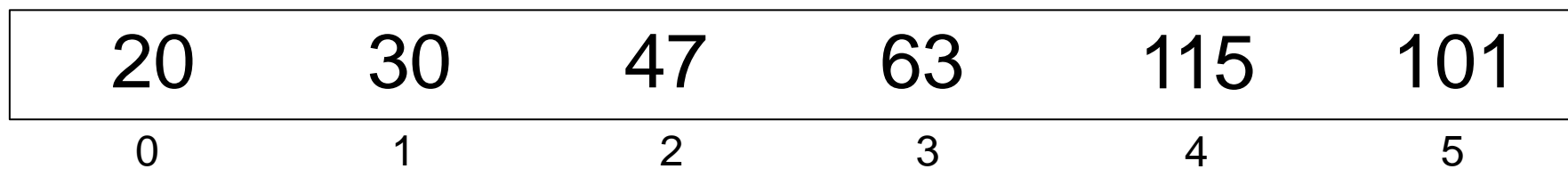
20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo

```

for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    int temp = array[menor];
    array[menor] = array[i];
    array[i] = temp;
}
    
```

false: $6 < 6$



Algoritmo

```
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
}
```

```
int temp = array[menor];
array[menor] = array[i];
array[i] = temp;
```

```
}
```

menor



i

20	30	47	63	101	115
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++) {  
        if (array[menor] > array[j]) {  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```



i

20	30	47	63	101	115
0	1	2	3	4	5

Algoritmo

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

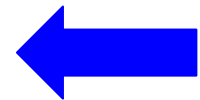
false: $5 < 5$



i

20	30	47	63	101	115
0	1	2	3	4	5

- Introdução sobre Ordenação Interna
- Funcionamento básico
- Algoritmo
- **Análise do número de movimentações e comparações**
- Conclusão



Análise do Número de Movimentações

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

Quantas
movimentações
(entre elementos
do *array*) são
realizadas?

Análise do Número de Movimentações

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++) {  
        if (array[menor] > array[j]) {  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

Quantas
movimentações
(entre elementos
do *array*) são
realizadas?

Análise do Número de Movimentações

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++) {  
        if (array[menor] > array[j]) {  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

O laço externo realiza $(n - 1)$ trocas, ou seja, $3(n - 1)$ movimentações

Quantas movimentações (entre elementos do *array*) são realizadas?

$$M(n) = 3(n - 1)$$

Análise do Número de Comparações

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    int temp = array[menor];  
    array[menor] = array[i];  
    array[i] = temp;  
}
```

Quantas
comparações
(entre elementos
do *array*) são
realizadas?

Análise do Número de Comparações

- No laço interno, sempre que procuramos o menor elemento entre i e n , fazemos $n-1$ comparações;
- No laço mais externo, estas $n-1$ comparações são feitas $n-1$ vezes;

Exemplo: $n = 5$

Para $i = 0$, os valores de j serão 1, 2, 3 e 4 = 4 vezes

Para $i = 1$, os valores de j serão 2, 3 e 4 = 3 vezes

Para $i = 2$, os valores de j serão 3 e 4 = 2 vezes

Para $i = 3$, o valor de j será 4 = 1 vez

Análise do Número de Comparações

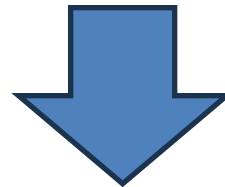
Exemplo: $n = 5$

Para $i = 0$, os valores de j serão 1, 2, 3 e 4 = 4 vezes = $(n-1)$

Para $i = 1$, os valores de j serão 2, 3 e 4 = 3 vezes

Para $i = 2$, os valores de j serão 3 e 4 = 2 vezes

Para $i = 3$, o valor de j será 4 = 1 vez



$$1 + 2 + 3 + \dots + (n-1) =$$

$$\sum_{k=1}^{n-1} k$$

Análise do Número de Comparações

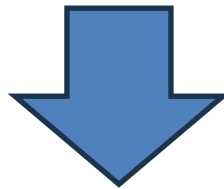
Exemplo: $n = 5$

Para $i = 0$, os valores de j serão 1, 2, 3 e 4 = 4 vezes = $(n-1)$

Para $i = 1$, os valores de j serão 2, 3 e 4 = 3 vezes

Para $i = 2$, os valores de j serão 3 e 4 = 2 vezes

Para $i = 3$, o valor de j será 4 = 1 vez




$$1 + 2 + 3 + \dots + (n-1) = \sum_{k=1}^{n-1} k = \frac{n(n-1)}{2} = \frac{n^2}{2} - \frac{n}{2} = \Theta(n^2)$$

- Assim, no total, o algoritmo faz $O(n^2)$ comparações, tendo o mesmo número de comparações tanto no pior caso, quanto no melhor caso;

Análise do Número de Movimentações

- Em relação ao número de atribuições, cada troca envolve 3 atribuições. Como são feitas $n-1$ trocas, temos $3(n-1) = O(n)$ atribuições de movimentação;
- Além destas, temos a atualização da posição do menor:
 - Esta ocorre em média $n \log n$ vezes;
 - Com estas, o custo total de atribuições é $O(n \log n)$;

- Introdução sobre Ordenação Interna
- Funcionamento básico
- Algoritmo
- Análise do número de movimentações e comparações
- **Conclusão** 

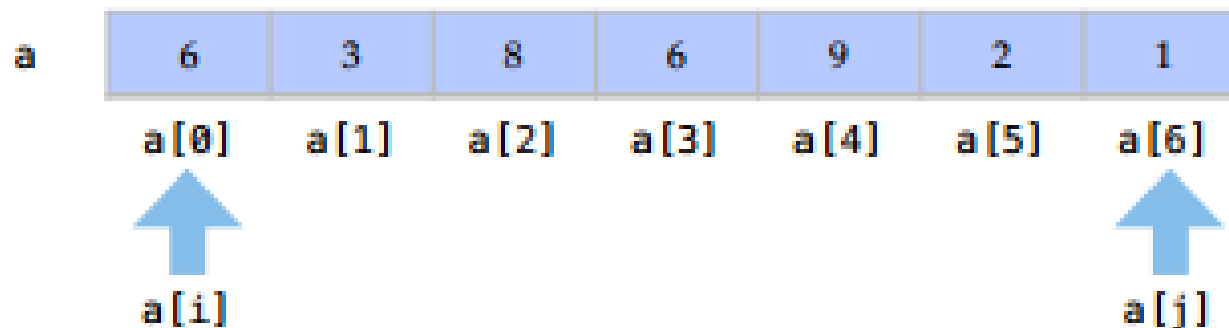
Conclusão

- Vantagens:
 - Em relação ao custo de movimentação, temos um custo linear $O(n)$;
 - É então um bom algoritmo onde estas movimentações por algum motivo custem muito;
 - O algoritmo só é interessante para arrays pequenos;

- Desvantagens:
 - Se o arranjo já estiver ordenado, isto não ajuda o algoritmo, pois o custo de comparações continua $O(n^2)$;
 - Quando isso ocorre, dizemos que o método não é adaptável

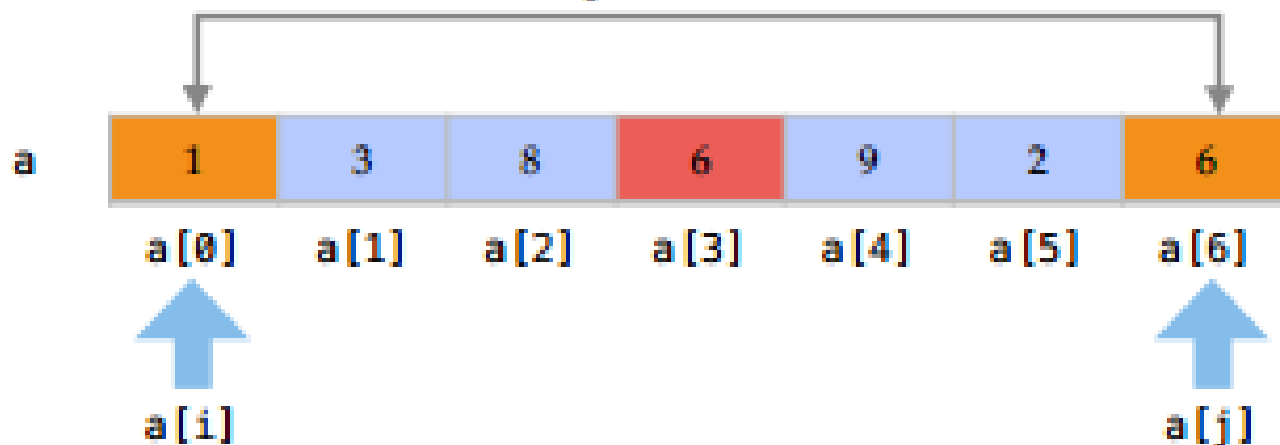
- Desvantagens:

- O algoritmo não é estável pois a troca entre elementos pode destruir a ordem relativa destes;
- Suponha um passo onde trocaremos o elemento $a[i]$ (6) com o elemento $a[j]$ (1):



- Desvantagens:

- Ao fazer esta troca, o elemento $a[i]$ (6) perde sua ordem relativa entre qualquer elemento entre $a[i+1]$ e $a[j-1]$;



- Neste caso, o elemento $a[i]$ perdeu sua ordem relativa com o elemento $a[3]$, que tem o mesmo valor

O algoritmo de seleção é *in place*?: